Design and Implementation of Online Experiments

nodeGame.org

Stefano Balietti

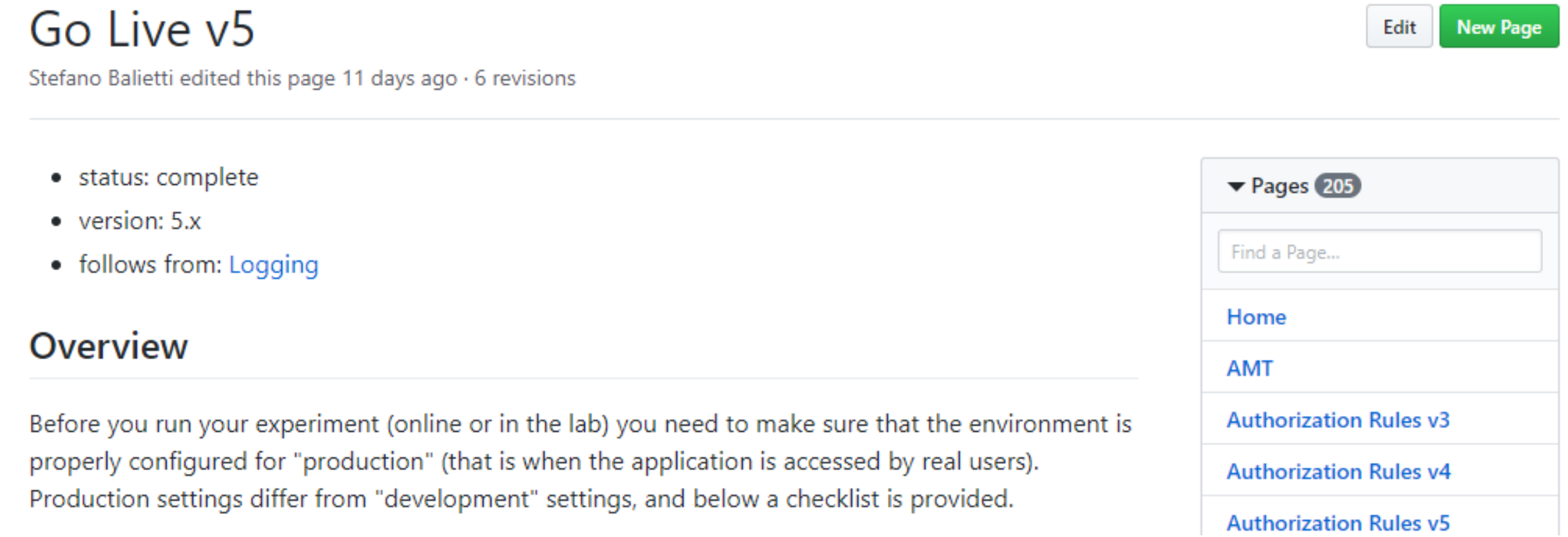
MZES and Heidelberg

Going Live With Your Experiment

@balietti
@nodegameorg
stefanobalietti.com@gmail.com

Going Live

Follow the checklist at the page: https://github.com/nodeGame/nodegame/wiki/Go-Live-v5
Try to follow as many rules as possible. At the minimum, those in the next slides.



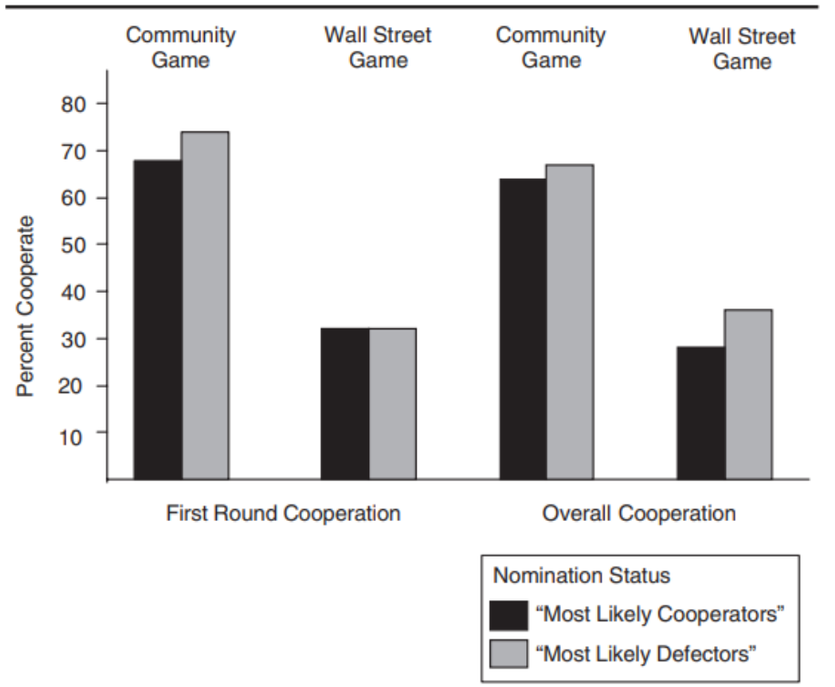
Hide the Game Name

The game name might be indicative of the research question at stake.

For example, if the name of the game contains the word "*sharing*," it might prime participants towards a more cooperative behavior.

The Name of the Game: Predictive Power of Reputations versus Situational Labels in Determining Prisoner's Dilemma Game Moves (2004)

Varda Liberman, Steven M. Samuels, Lee Ross



Hide the Game Name

1. Modify file **`public/js/index.js`**

```
// Connect to channel.
// (If using an alias if default channel, must pass the channel name
// as parameter to connect).
node.connect();
```

```
node.connect('/myexperiment');
```

The name of your game as specified when you created it (check file channel/channel.settings.js if you don't remember it).

2. Start the server with a default channel

```
node launcher.js --default=myexperiment
```

3. Open browser at: **`localhost:8080`**

Implement Authorization Rules

Modify file **`auth/auth.settings.js`**

When authorization is enabled, all players must connect through the address:

http://yourserver/yourgame/auth/id/password

This is address is different if you use the default option

where id and password are as you specified in your codes list.

In this way, you can make sure that only **authorized participants** can join once.

Implement Authorization Rules

Modify file **`auth/auth.settings.js`**

When authorization is enabled, all players must connect through the address:

http://yourserver/yourgame/auth/id/password

where id and password are as you specified in your codes list.

Authorized Codes

In this way, you can make sure that only ~~**authorized participants**~~ can join once.

Implement Authorization Rules

Modify file **`auth/auth.settings.js`**

When authorization is enabled, all players must connect through the address:

http://yourserver/yourgame/auth/id/password

where id and password are as you specified in your codes list.

Authorized Codes

In this way, you can make sure that only ~~**authorized participants**~~ can join once.

Why not a login page?

Most of online recruitment pools (e.g., MTurk, or MailChimp) work with a link

Implement Authorization Rules

Modify file **`auth/auth.settings.js`**

```
enabled: true,

//  - 'dummy': creates dummy ids and passwords in sequential order.
//  - 'auto':  creates random 8-digit alphanumeric ids and passwords.
//  - 'local': reads the authorization codes from a file. Defaults: codes.json, code.csv,
                or the value of the inFile option (available formats: json and csv).
mode: 'local',
```

Your live experiment should run with option set to **local**.
If you do not know how to create the authorization codes file, you could first set this option to "auto" and then changing it to "local".

Implement Requirement Rules

Modify file **`requirements/requirements.settings.js`**

- **speedTest:** is a test for both server and client. They should exchange messages fast enough. If the server is overloaded it will be slow. If the client has a bad connection it will also be slow. Notice, that the default value can be sometimes be too restrictive, you need to some tests and consider the distance from the server and your target population.

- **viewportSize:** to enforce a certain resolution (e.g., if your game needs a large UI)

- **browserDetect:** can enforce some browser requirements, as well as exclude mobile devices. When to exclude mobile users: if you use sliders, if your UI is very large, if you require to write long texts, in most group-experiments.

- **ES6Support:** if your code uses JavaScript ES6 syntax